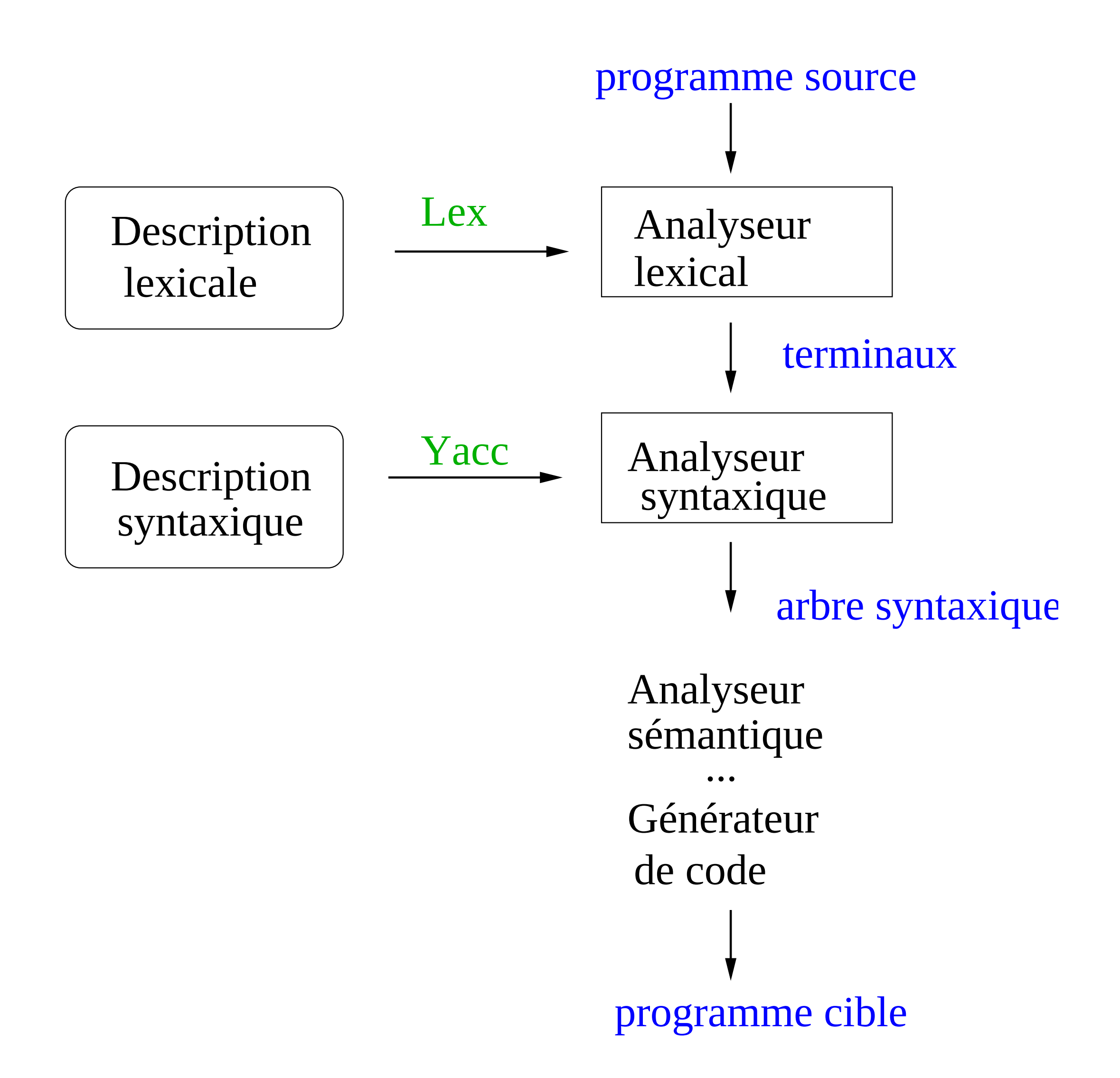
Introduction à Lex et Yacc

H. Cassé, M. Couzinier, M. Strecker

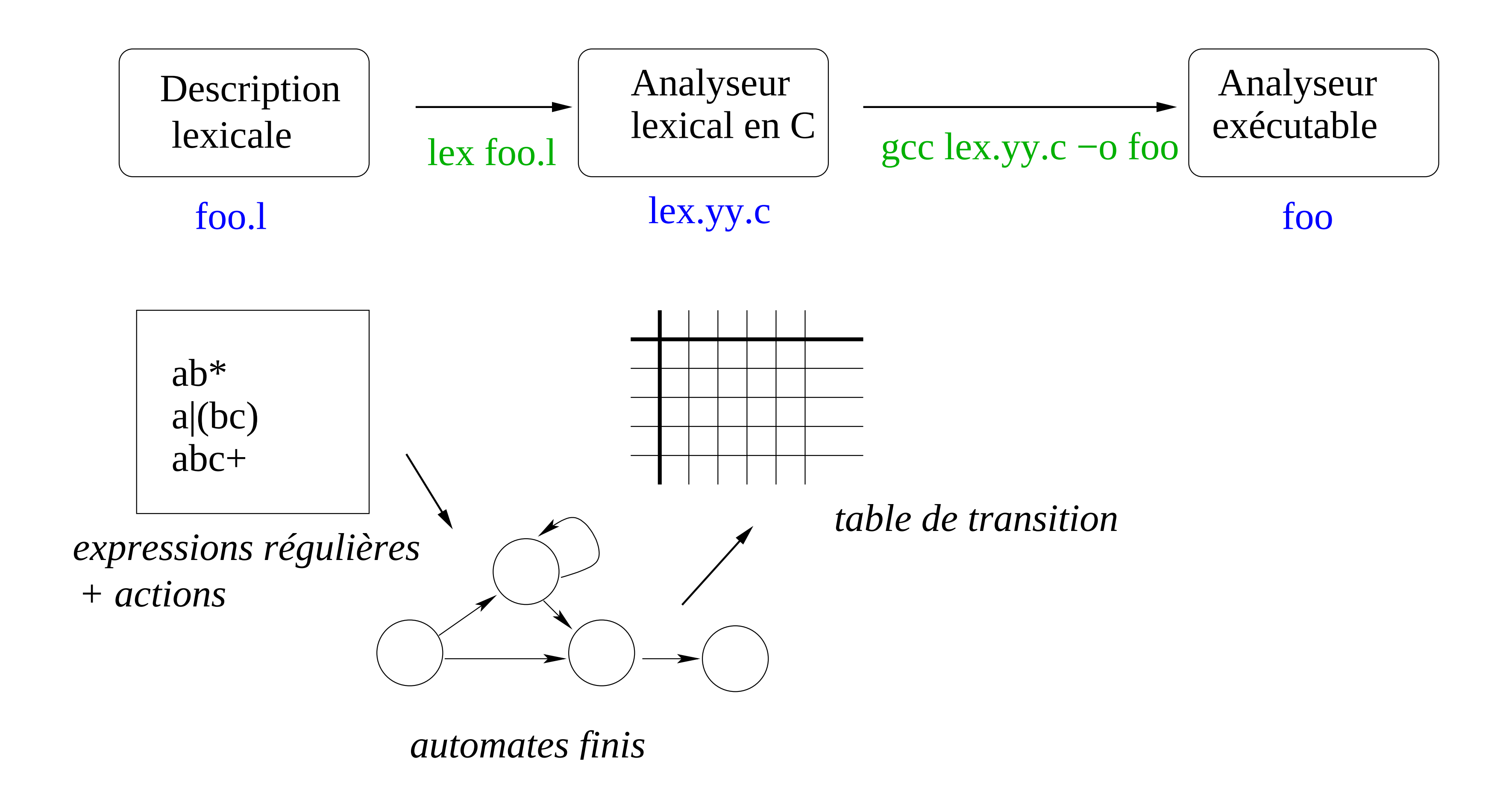
Année 2004/2005

1. L'analyseur lexical Lex
2. L'analyseur syntaxique Yacc
3. La coordination de Lex et Yacc

Processus de compilation



Fonctionnement de Lex



Syntaxe des expressions régulières

Caractères simples

`x`	le caractère `x`
`.`	(point) tout caractère sauf newline
`\n`	newline. Autres car. spéciaux: `\t`, `\r`

Classes de caractères

`[xyz]`	l'un des car. `x`, `y`, `z` (équivalent: `x|y|z`)
`[A-Z]`	les car. `A...Z`
`[^A-Z]`	tout car. sauf `A...Z`

Opérateurs

`rs`	concaténation
`r|s`	alternative
`r*`, `r+` `r{`n`}`	répétition (0 fois ou plus, 1 fois ou plus, n fois)

... et beaucoup plus. Regarder le manuel d'utilisation

Organisation de la description lexicale

Déclarations / définitions pour le programme C

```
%{
  int i;
%}
```

Abréviations d'expressions régulières

```
IDENT  [a-zA-Z][a-zA-Z0-9]*

%%
```

Expressions régulières et actions associées

```
{IDENT}";"   printf(...);

%%
```

Fonctions et programme principal

```
int main () {
...
}
```

Variables et fonctions prédéfinies de Lex

Variables:

`yyin`	fichier de lecture (défaut: `stdin`)
`yyout`	fichier d'ecriture (défaut: `stdout`)
`yytext`	dernière chaîne de caractère reconnue
`yyleng`	longueur de `yytext`

Fonctions:

`yylex()`	Appel de Lex, actif jusqu'au premier `return`
`yywrap()`	Pour traiter plusieurs fichiers. Ici: `return 1;`

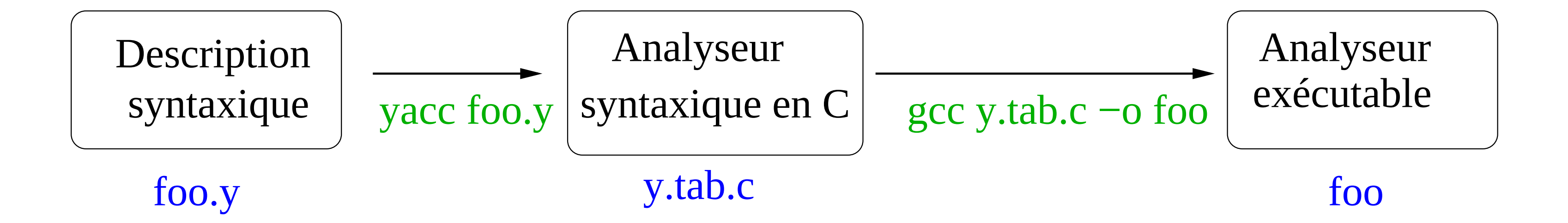
Lex: Exemple

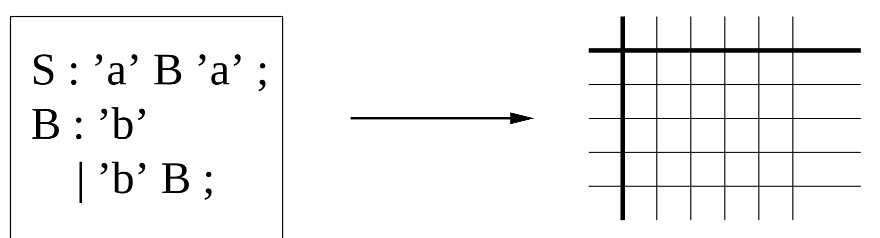
```
%{
        int num_lines = 0, num_chars = 0;
%}
%%
\n      ++num_lines; ++num_chars;
.       ++num_chars;

%%
main()
        {
        yylex();
        printf( "# of lines = %d, # of chars = %d\n",
                num_lines, num_chars );
        }
```

1. L'analyseur lexical Lex
2. L'analyseur syntaxique Yacc
3. La coordination de Lex et Yacc

Fonctionnement de Yacc





Organisation de la description syntaxique

Déclarations / définitions pour le programme C

```
%{
  int i;
%}
```

Déclaration de propriétés de symboles

```
%start S

%%
```

Règles de production de la grammaire et actions sémantiques

```
S : 'a' B 'a' { printf("..."); }
;
B : 'b'       { printf("..."); }
  | 'b' B     { printf("..."); }
;
%%
```

Fonctions et programme principal

```
int main () { ... }
```

Déclaration de propriétés de symboles

```
%union {
   déclaration C d'un champ d'union
}
```

Terminaux

```
%token<nom de champ> liste de terminaux
```

Non-terminaux

```
%type<nom de champ> liste de non-terminaux
```

Associativité des non-terminaux

```
%left liste de terminaux
%right liste de terminaux
```

Racine

```
%start non-terminal
```

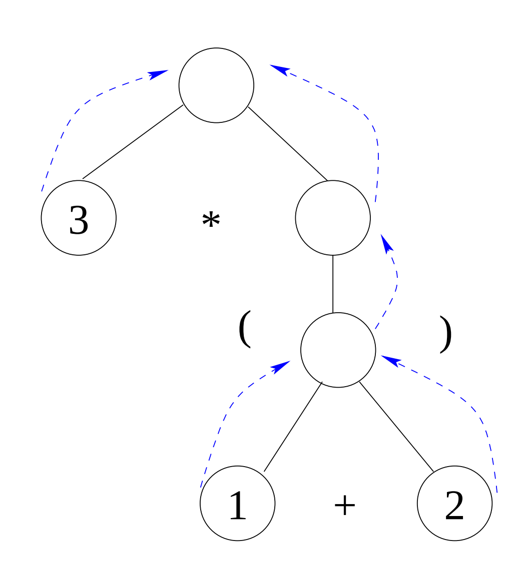
Actions sémantiques

En général: Commandes C comprises entre { ... }

```
B : 'b'        { printf("règle B1"); }
  | 'b' B      { printf("règle B2"); }
;
```

Accès aux sous-arbres:

```
E :  E '+' E    { $$ = $1 + $3; }
  |  ....
  | '(' E ')'   { $$ = $2; }
;
```



Conflits

Grammaire ambigüe:

```
S : 'a' B 'c'
|   'a' 'b' 'c'
;
B : 'b'
;
```

Invocation avec option `-v` crée fichier `y.output`

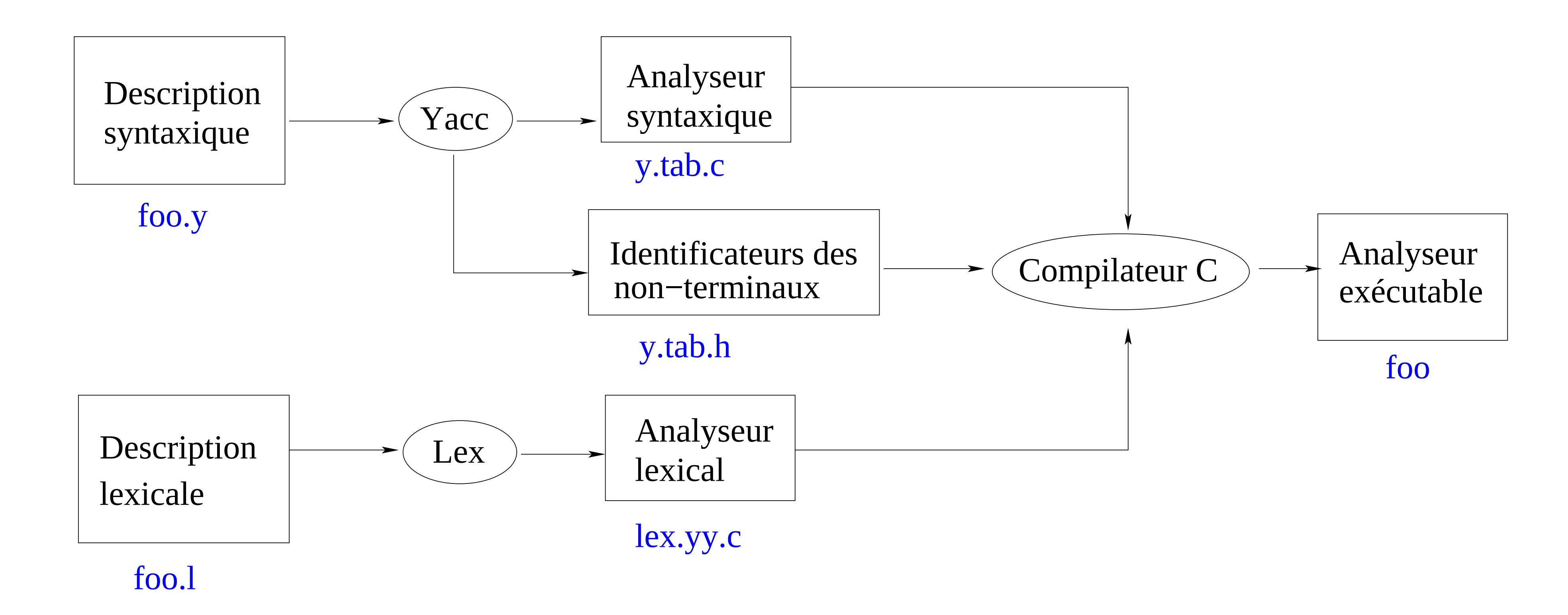
```
state 2
    S  ->  'a' 'b' . 'c'   (rule 2)
    B  ->  'b' .   (rule 3)

    'c'         shift, and go to state 4

    'c'         [reduce using rule 3 (B)]
```

1. L'analyseur lexical Lex
2. L'analyseur syntaxique Yacc
3. La coordination de Lex et Yacc

Schéma de compilation



Commandes:

```
> yacc -d foo.y
> lex foo.l
> cc y.tab.c lex.yy.c -ll -o foo
```

Exemple: Fichier Lex

```
%{
#include "y.tab.h"
#include <stdlib.h>
%}

BLANCS      [ \t\n]+
BOOLEAN     "T"|"F"
BINAIRE     [0-1]
OP          "^"

%%

{BLANCS}          /* On ne fait rien. */;
{BOOLEAN}         {yylval.Boolean=yytext;return BOOL;}
{BINAIRE}         {yylval.Binaire=atoi(yytext);return BIN;}
{OP}              {return OPERATOR;}
.                 {printf("erreur");}

%%
```

Exemple: Fichier Yacc (1)

Déclarations:

```
%{
#include <stdio.h>
#include <string.h>
%}
%union{
  char *Boolean;
  int Binaire;}

%token <Boolean> BOOL
%token <Binaire> BIN
%left  OPERATOR
%type <Binaire> term expr
```

Exemple: Fichier Yacc (2)

Grammaire:

```
%%
formule : expr {printf("valeur=%d\n",$1);}
;
expr : expr OPERATOR expr {$$ = $1 && $3;}
     | term {$$=$1;}
;
term : BOOL {if (!strcmp($1,"T")) $$= 1; else $$=0;}
     | BIN {$$= $1;}
;
```

Exemple: Fichier Yacc (2)

Fonctions:

```
%%
int yyerror(const char *msg) {
        printf("ERREUR: %s\n", msg);
        return 0;
}

extern FILE *yyin;

int main(void) {
        yyin = stdin;
        yyparse();
}
```